**BUFFER ISSUE RESOLUTION DOCUMENT (BIRD)**

**BIRD NUMBER:** 169

**ISSUE TITLE:** DLL Dependency Checking

**REQUESTOR:**  Arpad Muranyi, Mentor Graphics; Mike LaBonte, SiSoft

**DATE SUBMITTED:** June 24, 2014

**DATE REVISED:**

**DATE ACCEPTED BY IBIS OPEN FORUM:**

**STATEMENT OF THE ISSUE:**

IBIS-AMI models are often released with required DLLs missing, and as a result end users are unable to run simulations with the models without additional troubleshooting. What complicates matters is that the error messages are not always very helpful, which may make it hard for the users to determine why the model is not working.

This BIRD proposes the addition of a statement to the IBIS specification to remind model makers to check for any DLL dependencies before releasing their models to make sure that the recipients of the model will be able to use the models on the first attempt.

**ANALYSIS PATH/DATA THAT LED TO SPECIFICATION:**

**ANY OTHER BACKGROUND INFORMATION:**

Please note that the section quoted below is based on IBIS v6.0 plus the changes proposed by BIRD 155.2, which was accepted on October 11, 2013 by the IBIS Open Forum.

### 10.2.1 Overview

The executable model file of a Serializer-Deserializer (SERDES) transmitter or receiver contains up to five functions: “AMI\_Resolve”, “AMI\_Resolve\_Close”, “AMI\_Init”, “AMI\_GetWave” and “AMI\_Close”. The interfaces to these functions are designed to support three different phases of the simulation process: initialization, simulation of a segment of time, and termination of the simulation.

These functions (AMI\_Resolve, AMI\_Resolve\_Close, AMI\_Init, AMI\_GetWave and AMI\_Close) should all be supplied in a single executable model file, and their names and signatures must be as described in this section. The executable model files may reference other dynamic libraries as long as they are part of the operating system, or are delivered together with the executable model file by the model maker. To prevent failures related to dynamic library dependencies, models should be tested on each platform released using a dynamic library checking tool such as ldd or Dependency Walker on a “clean” system which has only an operating system installed that is compatible with the IBIS-AMI models being tested.

The five functions can be included in the executable model file in one of the following four combinations:

Case 1: Executable model file has AMI\_Init, AMI\_GetWave and AMI\_Close.

Case 2: Executable model file has AMI\_Init and AMI\_Close.

Case 3: Executable model file has AMI\_Resolve, AMI\_Resolve\_Close, AMI\_Init, AMI\_GetWave and AMI\_Close.

Case 4: Executable model file has AMI\_Resolve, AMI\_Resolve\_Close, AMI\_Init and AMI\_Close.”

Please note that the functions AMI\_Init and AMI\_Close are always required.

The interfaces to these functions are defined from three different perspectives. In addition to specifying the signature of the functions to provide a software coding perspective, anticipated application scenarios provide a functional and dynamic execution perspective, and a specification of the software infrastructure provides a software architecture perspective. Each of these perspectives is required to obtain interoperable software models.

Notes:

1. Throughout this section, terms “long”, “double” etc. are used to indicate the data types in the C programming language as published in ISO/IEC 9899-1999.
2. Throughout this section, text strings inside the symbols “<” and “>” should be considered to be supplied or substituted by the model maker.  Text strings inside “<” and “>” are not reserved and can be replaced.